

A parametric-complexity exact 3-COLORING Algorithm

JOSE ANTONIO MARTIN H., Complutense University of Madrid

This article presents a parametric-complexity exact algorithm to solve the graph 3-COLORING problem. Its maximal complexity is controlled by the parameter $\alpha \in \mathbb{N}$ which determines the running-time: $O(n^{f(\alpha)})$. The algorithm relies on the efficient search of “3-uncolorability witnesses” which, as it is shown here, is in $\text{NP} \cap \text{CoNP} \cap \text{P}$ for any fixed α , and in general for $\alpha \neq f(\text{input})$. The algorithm always returns either a legal 3-coloring, a 3-uncolorability witness or alternatively “undetermined”. Hence, it is not compulsory to trust neither the correctness of the algorithm itself nor the particular implementation used in order to be convinced that the obtained solution is correct. In the “undetermined” case, it can be re-run with a higher α until a solution is found, and so, a unique number $\alpha(G) \in \mathbb{N}$ corresponds to each graph G . Here, it is proved, for all $k \in \mathbb{N}$, that the proportion of graphs such that $\alpha(G) = k$ is greater than or equal to the proportion for $\alpha(G) > k$, hence the probability of $\alpha(G) > k$ has an upper-bound of $P(\alpha(G) > k) \leq 1/2^{k+1}$ and thus $\lim_{k \rightarrow \infty} P(\alpha(G) \leq k) = 1$. As a corollary, it is shown that as $\alpha \rightarrow \infty$ $(\text{NP} \cup \text{CoNP}) \setminus \text{P} \rightarrow \emptyset$ and that $\text{P} \neq \text{NP}$ implies $\alpha(G) \in [0, \infty)$. Hence, 3-colorability restricted to $\alpha(G) \leq k$ is in P . Nevertheless, it remains unknown whether it is NP -complete for some finite k . Experimental analyses have been thoroughly performed. The algorithm has been tested on planar graphs, 4-regular planar graphs and Erdős-Rényi random graphs. The experimental results validate the algorithmic implementation and confirm the theoretical expected results: for planar graphs, it was found that $\alpha = 1$ sufficed to obtain a solution in all the tests, while for random graphs, it has been observed that the distribution of $\alpha(G)$ obeys the given theoretical bounds.

Categories and Subject Descriptors: F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems – Computations on discrete structures

General Terms: Algorithms; Experimentation; Performance; Theory

Additional Key Words and Phrases: Computational complexity, Graph coloring, Parametric complexity

ACM Reference Format:

Martin H., J.A. 2011. A parametric-complexity exact 3-COLORING Algorithm. ACM 1, 1, Article 1 (July 2011), 26 pages.

DOI = 10.1145/0000000.0000000 <http://doi.acm.org/10.1145/0000000.0000000>

1. INTRODUCTION

Graph Coloring is one of the oldest and popular Constraint Satisfaction Problems (CSPs) [Jones 2008]. The study of efficient CSPs solving algorithms, and specifically for the graph coloring problem, is a central topic in Computer Science and Artificial Intelligence due to its wide applicability in many engineering projects, e.g., VLSI testing, planning and scheduling, timetabling, satellite range scheduling, register allocation, printed circuit testing and frequency assignment [Wigderson 1983; Park and Lee 1996; Ramani et al. 2004], as well as theoretical models, e.g., spin-glasses and the anti-ferromagnetic Potts model [Zdeborová and Krzakala 2007].

The problem of deciding if a given graph can be colored with k or less colors is called the k -colorability problem. For $k = 2$, the colorability problem can be solved efficiently.

Author's addresses: Jose Antonio Martin H. Department of Computer Architectures and Automatic Control, Faculty of Computer Science, Complutense University of Madrid.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© 2011 ACM 0000-0000/2011/07-ART1 \$10.00

DOI 10.1145/0000000.0000000 <http://doi.acm.org/10.1145/0000000.0000000>

However, for $k \geq 3$, in general, there is no known efficient algorithm and the problem is NP-complete¹ [Cook 1971; Karp 1972; Levin 1973; Garey and Johnson 1979].

Direct methods are just one interesting type among all the diverse methods used to approach NP-complete problems. They look for the determination of simple properties (e.g. triangle freeness) that impose necessary and sufficient conditions for determining the class of an instance (Yes or No). For the three-color-problem [Steinberg 1993] two examples are: phase-transition studies [Hogg et al. 1996; Culberson and Gent 2001; Mulet et al. 2002; Boettcher and Percus 2004; Zdeborová and Krzakala 2007], based on random graphs theory and sharp-thresholds [Erdős and Rényi 1960; Erdos et al. 1976], and pure structural-combinatorial approaches, based on, for example, the existence or absence of particular cycle configurations (see Steinberg 1993; Borodin 1996; Borodin et al. 2005; Wang and Chen 2007; Borodin et al. 2009). One of the foundational results along this approach is the Grötzsch's 3-color theorem [Thomassen 1994]: triangle-free planar graphs are 3-colorable.

On the other hand, given the intractability of the NP-complete problems, research on approximation algorithms started early, e.g., Johnson [1974a; 1974b], Garey and Johnson [1976], see also Wigderson [1983] and Berger and Rompel [1990]. The best known approximation guarantees for general 3-coloring was due to Halldórsson [1993], soon improved by Blum [1994] and more recently by Arora and Chlamtac [2006]. However, even for the approximate case graph coloring remains hard [Karger et al. 1998]. More strictly, it is NP-hard to even find a 4-coloring of a 3-chromatic graph [Khanna et al. 2000].

An alternative approach of more recent development, to the classical worst-case computational complexity theory, is Parameterized Complexity [Downey and Fellows 1999; Flum and Grohe 2006]. In Parameterized Complexity, apart from the problem instance itself, there is a parameter (usually an integer) that may be associated arbitrarily with the problem instance, allowing to study a problem's complexity with respect to both the size of the input (as in classical computational complexity) and the provided parameter. One of the main virtues of the parametric complexity approach is the concept of fixed-parameter tractability.

In this article a parametric-complexity exact algorithm, $\mathcal{A}(G, \alpha)$, to solve the graph 3-coloring problem is presented. Two key-differences with respect to past graph coloring approaches are that it is (1) exact and (2) polynomial-time (however parametric) while previous published algorithms are either low-exponential exact or polynomial-time (but approximate).

The algorithm introduced here relies on the search of 3-uncolorability witnesses. The definition of 3-uncolorability witness presented here is (to the best of my knowledge) the first one that is formally defined and the most naturally related to the 3-coloring problem. A 3-uncolorability witness is defined² as a sequence of "forced vertex contractions" leading to a graph containing a K_4 . A verifier can check that every contraction is "forced" efficiently. The possibility of efficiently generating 3-uncolorability witnesses is of theoretical interest since 3-colorability is NP-complete and thus 3-uncolorability is CoNP-complete.

Some works have studied short proofs systems for CoNP-complete problems (e.g. Boppana et al. 1987; Fortnow and Sipser 1988). In particular, for graph coloring, there is a recent work [Bes and Jegou 2005] that looks for proving graph uncolorability with

¹The computational complexity terminology and concepts (e.g. P, NP, CoNP, NP-completeness, NP-Hardness, certificate/witness, reductions) can be consulted in the recent books of Arora and Barak [2009] and Goldreich [2008].

²A 3-uncolorability witness can also be initially understood as a (pruned) Zykov-tree (see Corneil and Graham 1973) in which every leaf is a graph containing a K_4 subgraph.

consistency checks of constraint satisfaction problems. However, short 3-uncolorability witnesses are not possible in general unless $\text{NP} = \text{CoNP}$, which is unknown and indeed unlikely.

The proposed $\mathcal{A}(G, \alpha)$ algorithm has several “good” key-features:

- (1) The algorithm’s complexity is controlled by means of a simple parameter (α , the maximum recursion level). The parameter determines the order of the polynomial that bounds the running-time of the algorithm: $O(n^{f(\alpha)})$.
- (2) The algorithm generates witnesses for both Yes and No instances, i.e., either a legal 3-coloring or a 3-uncolorability witness.
- (3) For any fixed α , and in general for $\alpha \neq f(\text{input})$, finding a 3-uncolorability witness or a legal 3-coloring is in: $\text{NP} \cap \text{CoNP} \cap \text{P}$.
- (4) If for a given α the algorithm is unable to find a witness then it returns “undetermined” so that it can be re-run with a higher α up to a solution is obtained, thus, to each input graph G , the algorithm assigns a unique α , i.e., $\alpha = f(G)$.
- (5) The possibility of self-tuning the α parameter (by using the undetermined return value) gives the algorithm the capability of using only the required computational resources for a particular problem instance, e.g., for almost all planar graphs it is enough a value of $\alpha = 0$ to obtain the right solution, obtaining thus an $O(n^2)$ algorithm for the 3-coloring problem in almost all planar graphs.
- (6) Also, apart from computational complexity results, a very important feature of the algorithm is that it gives stand-alone indubitable results and therefore it is not necessary to trust the correctness of the algorithm itself, nor the particular implementation used, to recognize that the provided solution is correct, since the verification of the result can be done efficiently using just only the solution provided (the witness or certificate).

Moreover, from the theoretical point of view, the most important property of the algorithm is the classification of all random graphs by the number $\alpha(G)$: the minimum value of the parameter α , for which the algorithm is able to obtain a witness, given a particular Yes/No instance G of the 3-coloring problem. This brings important consequences and allows the development a more thorough analysis:

THEOREM 1.1. *Let G be a random graph, if $P(\alpha = k)$ is the probability that $\alpha(G) = k$ and $P(\alpha > k)$ is the probability that $\alpha(G) > k$, for all $k \in \mathbb{N}$, then:*

$$P(\alpha = k) \geq P(\alpha > k) \quad (1)$$

COROLLARY 1.2. *If G is a random graph and $P(\alpha > k)$ is the probability that $\alpha(G) > k$, for all $k \in \mathbb{N}$, then:*

$$P(\alpha > k) \leq \frac{1}{2^{k+1}}; \quad (2)$$

$$\lim_{k \rightarrow \infty} P(\alpha \leq k) = 1 \quad (3)$$

COROLLARY 1.3. *if α is the parameter of the $\mathcal{A}(G, \alpha)$ algorithm then:*

$$\text{as } \alpha \rightarrow \infty \quad (\text{NP} \cup \text{CoNP}) \setminus \text{P} \rightarrow \emptyset \quad (4)$$

Finally:

LEMMA 1.4. *If $\text{P} \neq \text{NP}$ then $\alpha(G) \in [0, \infty)$,*

which means that: if $\alpha(G)$ is finite, i.e., $\alpha(G) \in [0, n]$ for some $n \in \mathbb{N}$, then $\text{P} = \text{NP}$.

Even assuming the infiniteness of $\alpha(G)$, the problem now is not whether a polynomial algorithm exists or not, but instead, to determine whether 3-colorability for a class of graphs with $\alpha(G) \leq n$ is NP-complete.

1.1. Experimental analysis

The problem of evaluating algorithms experimentally could be very tricky if tests are performed on “artificial instances”, which may be uncorrelated or isolated from any specific practical application. Nevertheless, there are some lines of research suggesting special distributions of graph instances on which purported NP-complete problem solvers should be evaluated to determine properly their performances (see Selman et al. 1996; Culberson and Gent 2001; Mizuno and Nishihara 2008). Moreover, apart from the “harder instances approach”, Johnson [2002] has proposed a more complete methodological approach to the experimental analysis of algorithms.

In the experimental part of this article, the algorithm is thoroughly evaluated over significant samples pertaining to three different graph distributions:

- (1) Pseudo-random planar graphs [Denise et al. 1996; Bodirsky et al. 2003].
- (2) Random 4-regular planar graphs [Manca 1979; Lehel 1981; Broersma et al. 1993].
- (3) Erdős-Rényi connected random graphs [Erdős and Rényi 1960].

Each class, and each distribution, has a good justification:

(1) Planarity imposes some interesting structural properties, i.e., the three color problem on planar graphs is the only unqualified problem that remains open [Steinberg 1993] since 1-coloring is trivial, 2-coloring is well characterized and the maximum chromatic number on the plane is four [Appel et al. 1977a; Appel et al. 1977b], and at the same time determining 3-colorability of planar graphs is NP-complete [Stockmeyer 1973; Garey and Johnson 1979].

(2) Even more, 3-colorability of four-regular planar graphs still remains NP-complete [Dailey 1980] and most important, in this class the average degree is fixed so the phase-transition phenomenon as defined for random graphs could not be applied directly in this case.

(3) Finally, sampling from the Erdős-Rényi (connected) random graphs distribution gives the necessary theoretical support for evaluating an algorithm in the general case, validating the theoretical bounds and allowing to obtain results that could be compared against other algorithms in the literature, e.g., the best performing 3-coloring algorithms proposed in the literature are surveyed by Malaguti and Toth [2010].

An interesting experimental finding is that in all the test cases for planar graphs, it was found that fixing the maximum recursion level to $\alpha = 1$ was enough to obtain a solution, that is, exact efficiently verifiable results were obtained with a polynomial algorithm. This was also the case for 4-regular planar graphs.

Furthermore, in the general 3-coloring case (random graphs) experiments, it has been observed that the distribution of $\alpha(G)$ conforms to the theoretical decreasing pattern: the majority of graphs are in $\alpha(G) = 0$ or $\alpha(G) = 1$, some in $\alpha(G) = 2$, a few in $\alpha(G) = 3$, very few in $\alpha(G) = 4$ and so on. However, it was not possible to obtain a graph with $\alpha(G) > 4$ in the sampled random graphs.

As observed in the experiments, the value of $\alpha(G)$ is directly correlated with the average degree $d = 2m/n$ (m = edges, n = vertices), so near the phase transition threshold (d^*) the probability of a higher value of $\alpha(G)$ increases.

$$d^* \approx \begin{cases} 4.69, & \text{Mulet et al. [2002];} \\ 4.703, & \text{Boettcher and Percus [2004];} \end{cases} \quad (5)$$

However, many interesting questions remain open, e.g.:

- What is the exact distribution of $\alpha(G)$ in random graphs?
- Apart from the average degree, what other parameters are related to $\alpha(G)$?

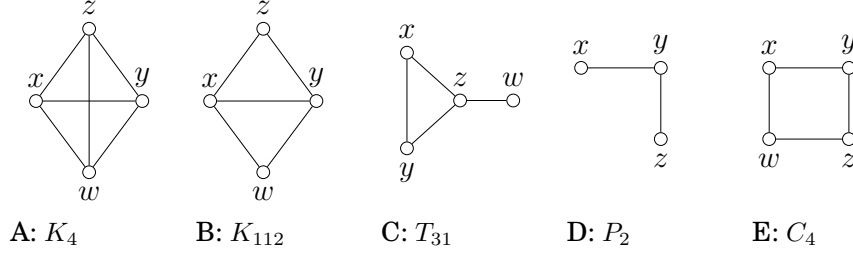


Fig. 1: (A) the complete graph K_4 . (B) a complete 3-partite graph K_{112} . (C) a tadpole graph T_{31} which imposes a binary constraint on 3-coloring: either G/xw or G/yw . (D) the path of length two P_2 . (E) a C_4 graph that imposes a binary constraint on 3-coloring: either G/xz or G/yw .

- Given an arbitrary input graph G , can it be predicted (exactly or approximately) the $\alpha(G)$ value and what is the best possible approximation to $\alpha(G)$?
- As for the chromatic number $\chi(G)$, is there any graph construction mechanism that allows the generation of graphs with arbitrarily large $\alpha(G)$?

1.2. Preliminary definitions, terminology and notation

This article follows the standard graph theory terminology. For general terms and notation the book of Jensen and Toft [1995] and the recent book, on chromatic graph theory, by Chartrand and Zhang [2008] should be consulted. However, some special terms and specific notations are defined next.

Unless we state it otherwise, all graphs in this work are connected and simple (finite, and have no loops or parallel edges). A graph is called *planar* if it can be drawn in a plane in such a way that its edges intersect only in its end-points, i.e., there are no edges crossing.

We refer to u, v as a *planar preserving edge* if uv is an edge of the complement of G , $uv \in \bar{G}$, and $G + uv$ remains planar.

Sets are specified using $\{a, b, c\}$ notation and $H \subset G$ or $H \subseteq G$ reads H is a proper subgraph of G or a subgraph of G respectively. A *vertex contraction*, also called vertex identification or vertex merging, is denoted by G/uv . Vertex or edge addition and deletion are denoted as: $G + u$ or $G + uv$ and $G - u$ or $G - uv$ respectively.

The special named graphs used in the article are: the complete graph K_4 , the complete 3-partite graph K_{112} (a K_4 minus one vertex), the *tadpole* graph T_{31} obtained by joining a triangle T_3 to a single vertex with a bridge, the *triangle* graph T_3 , consisting of three fully connected vertices, e.g., $\{x, y, z\}$ such that xy, yz, xz are edges, the path graph P of length two P_2 and the square graph C_4 which is a cycle of length four.

2. DEFINITIONS AND THEORY

The algorithm introduced here relies heavily on Grötzsch's theorem since the key procedures are based on triangle (or triangle configurations) processing to search for a 3-uncolorability witness.

A *witness* (or *certificate*, Arora and Barak [2009]) is an efficiently verifiable proof of the correctness of an answer for some given decision problem. For instance, given a graph G , a legal 3-coloring of G or a short-proof of the uncolorability of G with only three colors are witnesses for the 3-colorability problem.

Definition 2.1. Given a graph G and a 3-colorable subgraph H of G , there is a *forced vertex contraction*³ on $\{u, v\} \in V(G)$ if adding the edge uv to H makes H not 3-colorable.

Definition 2.2. Given a non-3-colorable input graph G , a *3-uncolorability witness* \mathcal{W} is a description of a (possibly empty) sequence of vertex contractions, G/uv , leading to a graph containing K_4 , such that either:

- (1) u, v are the non-complete vertices of a complete 3-partite K_{112} subgraph of G or;
- (2) a nested 3-uncolorability witness for the graph $G + uv$ is provided.

THEOREM 2.3. *Finding a 3-uncolorability witness is in: $NP \cap CoNP \cap P$; for any fixed recursion level $\alpha = k \in \mathbb{N}$.*

PROOF.

- (1) The problem is in NP:
 - **Certificate:** the 3-uncolorability witness (\mathcal{W}) itself.
 - **Polynomial Certifier:** check if each element of \mathcal{W} is a forced vertex contraction and perform it up to obtaining a K_4 subgraph. If the recursion level α is constant then the size of the witness is bounded by a polynomial:

For a recursion level of $\alpha = 0$, meaning that \mathcal{W} contains only vertex contractions based on K_{112} subgraphs and no other nested witnesses, the size of \mathcal{W} is bounded by the maximum number of edges of G which is of order $O(n^2)$. Assuming that, for every constant $\alpha = k$, it is bounded by a polynomial P^k then we can specify the order of a recursion level of $\alpha = k + 1$ as $P^k \times O(n^2)$, which is also a polynomial on $n = |V(G)|$, since every nested witness is of order $O(n^2)$.
- (2) Moreover, 3-uncolorability is the complementary problem of graph 3-colorability, an NP-complete problem. Hence 3-uncolorability is CoNP-complete.
- (3) That it is also in P for a fixed α follows from the existence of a polynomial algorithm to solve it.

□

Now, let us define a greedy 3-coloring algorithm that will serve as a baseline from where to derive the proposed coloring algorithm.

Definition 2.4. The $g_3(G)$ algorithm is a “greedy” 3-coloring algorithm that sequentially, and at each step, selects two non-adjacent vertices x and y of a graph G and contracts them to obtain the graph G/xy , maintaining a list S of the vertices that have been contracted so far, so that if in S are just only three independent sets then these are three color classes of G and hence a legal 3-coloring of G .

The proposed parametric-complexity exact algorithm will use the same approach to 3-coloring as the following lemma below (lemma 2.5):

LEMMA 2.5. *Given an exact algorithm $\mathcal{W}(G)_0$ of complexity $O(n^k)$ to obtain a 3-uncolorability witness for any non-3-colorable graph then there is an exact algorithm $\mathcal{W}(G)_1$ of complexity $O(n^{k+1})$ to obtain a 3-coloring of any 3-colorable graph.*

³Two vertices of a graph are said to be 3-chromatically connected if they are assigned the same color in any 3-coloring of the graph [Steinberg 1993]. Culberson and Gent [2001] use instead the term “frozen pair” and define the spine of a colorable graph G as: $B(G) = \{\{u, v\} : \{u, v\} \text{ is frozen by } G\}$, as we can see, there is a direct relation between $B(G)$ and the definition of the 3-uncolorability witness given here. The same concept for such a pair $\{u, v\}$ is also defined as an implicit-identity by the current author who gives [Martin H. 2011] a more thorough study of this subject for the general case (k-chromatic and non-planar graphs).

PROOF. Assume $\mathcal{W}(G)_0$ exists.

Then, given a 3-colorable graph G , apply the greedy $g_3(G)$ algorithm but avoiding the contraction of every $\{x, y\}$ such that G/xy is not 3-colorable, which can be determined in $O(n^k)$ by $\mathcal{W}(G/xy)_0$. Since G is 3-colorable this will converge to a triangle graph. Since $g_3(G)$ is of complexity $O(n)$ (at every step at least one vertex gets colored) then we obtain a 3-coloring of G in $O(n^{k+1}) = O(n) \cdot O(n^k)$ \square

COROLLARY 2.6. *So, if $\mathcal{W}(G, \alpha)_0$ is an exact parametric algorithm, of complexity $O(n^{f(\alpha)})$, to obtain a 3-uncolorability witness for any non-3-colorable graph then there is an exact parametric algorithm $\mathcal{W}(G, \alpha)_1$ of complexity $O(n^{f(k+1)})$ to obtain a 3-coloring of any 3-colorable graph.*

To formalize the analysis of the algorithm let us define the following two algorithm specifications:

Definition 2.7. The $\mathcal{A}(G, \alpha)$ is a parametric-complexity algorithm that computes a function that assign to a given input graph G just one of three possible values: 0, 1 or ∞ , when G is respectively: non-3-colorable, 3-colorable or the algorithm was unable to find a solution for the given value of the α parameter:

$$\mathcal{A}(G, \alpha) \begin{cases} 0, & \text{No-instance: } G \text{ is not 3-colorable;} \\ 1, & \text{Yes-instance: } G \text{ is 3-colorable;} \\ \infty, & \text{undetermined for the given } \alpha \end{cases} \quad (6)$$

Definition 2.8. The $\mathcal{W}(G, \alpha)$ is a parametric-complexity algorithm that computes a function that assigns to a given input graph G just one of three possible values: a 3-uncolorability witness, a legal 3-coloring or a null value, when $\mathcal{A}(G, \alpha)$ is respectively: 0, 1 or ∞ .

$$\mathcal{W}(G, \alpha) \begin{cases} \text{a 3-uncolorability witness, } \mathcal{A}(G, \alpha) = 0; \\ \text{a legal 3-coloring, } \mathcal{A}(G, \alpha) = 1; \\ \emptyset, \mathcal{A}(G, \alpha) = \infty \end{cases} \quad (7)$$

Definition 2.9. Given a graph G , the α level of G : $\alpha(G)$, is the minimum α required to obtain a witness for a particular instance of the 3-(un)colorability problem:

$$\alpha(G) = \min(\alpha) \quad \text{s.t.} \quad \mathcal{W}(G, \alpha) \neq \emptyset \quad (8)$$

2.1. Proof of the main theorem and corollaries

Let us define the number $N(\phi(\alpha))$ as the cardinality of the set of all graphs that satisfy some logical function, $\phi(\alpha)$, of α , i.e.,:

$$N(\phi(\alpha)) = |\{G \in G^* \mid \phi(\alpha)\}|, \quad (9)$$

where G^* is the set of all graphs.

The key argument of the proof uses the assertion of lemma 2.5 that it suffices to be able to solve 3-uncolorability for some α in order to be able to solve 3-colorability for the same α . To prove the theorem the following two auxiliary lemmas need to be proved first:

LEMMA 2.10. *Almost all graphs have $\alpha(G) = 0$.*

PROOF. As proved by Erdos et al. [1976] and improved by Kolaitis et al. [1987], almost all $K_{\ell+1}$ -free graphs are ℓ -colorable and hence almost all K_4 -free graphs are 3-colorable, i.e., if G is a K_4 -free random graph then with high probability it is 3-colorable. Thus, if we assume that G is not 3-colorable then with high probability it is not K_4 -free, that is, almost all non-3-colorable graphs contain a K_4 . So, almost all

non-3-colorable graphs have $\alpha(G) = 0$. Then, by lemma 2.5, for almost all 3-colorable graphs a legal 3-coloring can be found also with $\alpha = 0$ and thus almost all 3-colorable graphs also have $\alpha(G) = 0$. Therefore almost all graphs have $\alpha(G) = 0$. \square

LEMMA 2.11. *If for some $k \in \mathbb{N}$ almost all graphs have $\alpha(G) = k$ and almost all graphs with $\alpha(G) > k$ can be transformed into a graph with $\alpha(G) = k$, by applying a series of random edge additions or vertex contractions, then almost all graphs with $\alpha(G) > k$ have $\alpha(G) = k + 1$.*

PROOF. Before starting the proof, it is useful to show first that $\alpha(G) > k$, for all $k \in \mathbb{N}$, is a monotone property⁴ (see Alon et al. 2010).

The key-idea of the proof is that, with high probability, by the application of a minimal series of random edge additions and vertex contractions, almost all graphs with $\alpha(G) = k + 1$ are transformed into a graph with $\alpha(G) = k$. And thus show by induction that:

- (1) almost all graphs with $\alpha(G) > 0$ have $\alpha(G) = 1$
- (2) almost all graphs with $\alpha(G) > k$ have $\alpha(G) = k + 1$
- (3) almost all graphs with $\alpha(G) > k + 1$ have $\alpha(G) = k + 2$

The proof is by induction on k , starting at $k = 0$. Almost all graphs have $\alpha(G) = 0$. The class of graphs with $\alpha(G) = 0$ is the class of K_4 -free graphs, even-wheels-free graphs and in general every witness that collapses to a K_4 only by the identifications of K_{112} subgraphs. Since the number of graph transformation operations on a graph G in order to obtain a graph H with $\alpha(H) = 0$ is a monotonically increasing function of $\alpha(G)$ then almost all graphs with $\alpha(G) > 0$ have $\alpha(G) = 1$ since the probability of transforming a graph with $\alpha(G) = 1$ into a graph with $\alpha(H) = 0$, with fewer operations than for $\alpha(G) > 1$, is bigger than for graphs with $\alpha(G) > 1$.

So, assume it holds for every k , that is, almost all graphs with $\alpha(G) > k$ have $\alpha(G) = k + 1$, and show that then it holds for $k + 1$. Let us suppose that almost all graphs with $\alpha(G) > k + 1$ have $\alpha(G) > k + 2$. Then, among all the graphs with $\alpha(G) > k + 2$ the probability of obtaining a graph with $\alpha = k$ is bigger than the probability of obtaining a graph with $\alpha(G) = k$ for graphs with $\alpha(G) = k + 1$, which contradicts the inductive steps since almost all graphs with $\alpha(G) > k$ have $\alpha(G) = k + 1$. \square

Hence, by the lemma 2.11, the theorem 1.1 holds for all k since:

- (1) By the lemma 2.10: $N(\alpha = 0) \geq N(\alpha > 0)$.
- (2) For all k , almost all graphs with $\alpha(G) > k$ have $\alpha(G) = k + 1$ implies:

$$N(\alpha = k) \geq N(\alpha > k),$$

- (3) Therefore, it follows for random graphs that:

$$P(\alpha = k) \geq P(\alpha > k), \quad \forall k \in [0, \infty)$$

This completes the proof. \square

Statements of Corollary 1.2 follows immediately from $N(\alpha = k) \geq N(\alpha > k)$. Corollary 1.3 is proved next:

PROOF. of Corollary 1.3

Let $\text{NP}(\alpha)$ and $\text{CoNP}(\alpha)$ be respectively:

- (1) the set of all problem instances in NP which are in P for a given α .
- (2) the set of all problem instances in CoNP which are in P for a given α .

⁴A monotone property of a graph G is a property that holds for G and also for every arbitrary subgraph of G , e.g., K_ℓ -free graphs.

$$\text{NP}(\alpha) = \{E \in \text{NP} : \forall G \in E; \mathcal{A}(G, \alpha) \neq \infty\}; \quad (10)$$

$$\text{CoNP}(\alpha) = \{E \in \text{CoNP} : \forall G \in E; \mathcal{A}(G, \alpha) \neq \infty\}, \quad (11)$$

where E is the class of 3-coloring instances (G) which are solvable in polynomial-time by the algorithm $\mathcal{A}(G, \alpha)$ for some particular value of α .

Then, since 3-(un)colorability witness is in $\text{NP} \cap \text{CoNP} \cap \text{P}$, for any fixed α , then $\text{NP}(\alpha) = \text{CoNP}(\alpha)$ and so $(\text{NP}(\alpha) \cup \text{CoNP}(\alpha)) \setminus \text{P} = \emptyset$.

It is clear then that as $\alpha \rightarrow \infty$ $\text{NP}(\alpha) \rightarrow \text{NP}$ and $\text{CoNP}(\alpha) \rightarrow \text{CoNP}$, hence: as $\alpha \rightarrow \infty$ $(\text{NP}(\alpha) \cup \text{CoNP}(\alpha)) \setminus \text{P} \rightarrow (\text{NP} \cup \text{CoNP}) \setminus \text{P}$ and hence $(\text{NP} \cup \text{CoNP}) \setminus \text{P} \rightarrow \emptyset$

This completes the proof. \square

Finally, a proof of lemma 1.4 is direct since a finite $\alpha(G)$ implies $\text{P} = \text{NP}$.

3. DESCRIPTION OF THE ALGORITHM

The algorithm is divided in two parts: the decision problem (`is_3colorable`) and the coloring algorithms (`_3COL`). There are two versions of the coloring algorithms: one for planar graphs (`planar_3COL`) and one for non-planar graphs (`general_3COL`). This last division is oriented to take advantage of some special structural constraints of planar graphs (e.g. Grötzsch's like theorems) that aid the development of more efficient algorithms. First, the algorithm for the planar graphs case is described, which is better for understanding the key-idea behind the algorithms, then it is generalized to the non-planar graphs case.

3.1. The decision routine

In simple terms, the decision algorithm consists in finding (or building) a 3-uncolorability witness, subject to a given α level.

The definition (2.2) of uncolorability witness involves a recurrent term closely related to the Zykov [1952] theorem:

$$\chi(G) = \min(\chi(G/uv), \chi(G + uv)), \quad (12)$$

defined for the pair $\{u, v\}$ of non-adjacent vertices.

Hence, we can determine 3-colorability by searching along the tree defined by the following recurrence relation:

$$\chi(G) \leq 3 \iff \min(\chi(G/uv), \chi(G + uv)) \leq 3 \quad (13)$$

In the proposed algorithm the height of the tree is bounded by α and the selection of the uv 's follows special priority rules, i.e., contract first every $\{u, v\}$ belonging to every K_{112} subgraph. These priority rules will define the search tree and so its size (number of nodes). Therefore, the parametric algorithm follows the breadth-first search strategy.

Although the size (but not the tree itself) of every Zykov-tree of a given graph G is the same, independently of the selected way of branching, the situation is different for partial (pruned) Zykov-trees [McDiarmid 1979]. For instance, the use of K_{112} subgraphs, which are the special subgraphs that involves the essence of the uncolorability witnesses, reduce an exponential growing subtree into a linear one by eliminating the $G + uv$ branch, allowing to obtain smaller witnesses.

Hence, given a graph G and $\alpha = 0$:

- (1) The algorithm will find every K_{112} subgraph of G and contract its non-complete vertices.
- (2) If in doing this a K_4 subgraph is produced then there is a 3-uncolorability witness.
- (3) If in doing this a K_3 subgraph is produced then there is a legal 3-coloring.
- (4) If all K_{112} subgraphs have been processed so that G does not contain neither a K_{112} nor a K_4 subgraph then return "undetermined".

The algorithm described above is the base case ($\alpha = 0$) of the decision algorithm. Nevertheless, it is not exact. The next step is to define it for every α to obtain an exact algorithm.

Given a value of $\alpha = k$, the method consist in processing every T_{31} tadpole subgraph (x, y, x, w) of G in the following way:

- (1) Test base case $\alpha = 0$ and return any possible 3-uncolorability witness.
- (2) Select an unvisited T_{31} tadpole subgraph (x, y, x, w) of G .
- (3) Test the vertex contraction G/xw and apply the test for $\alpha = k - 1$.
- (4) If the algorithm obtains a 3-uncolorability witness for G/xw then working with the graph $G + xw$ is mandatory and so $\{y, w\}$ is a forced vertex contraction based on a nested 3-uncolorability witness, since $G + yw$ will lead to a K_4 .
- (5) If in doing this process a K_4 subgraph is produced then return 0 and the 3-uncolorability witness.
- (6) If all T_{31} tadpole subgraphs have been visited and no contraction has occurred then return ∞ (undetermined). Otherwise continue at step (1) with the current graph.

In the reference implementation, the main routine of the coloring algorithm has been named: `planar_3COL` while the decision routine is named `is_3colorable` (cf. Algorithm 1).

The decision routine `is_3colorable` returns one of three possible results:

$$\text{is_3colorable} = \begin{cases} \langle 0, \mathcal{W}_0 \rangle, & \text{a 3-uncolorability witness } \mathcal{W}_0 \text{ was found;} \\ \langle 1, \mathcal{W}_1 \rangle, & \text{a 3-coloring } \mathcal{W}_1 \text{ was found;} \\ \langle \infty, \emptyset \rangle, & \text{undetermined for the given } \alpha \end{cases} \quad (14)$$

Algorithm 1 `is_3colorable(G, α)`

```

1:  $N \leftarrow |V(G)| + 1$ 
2: while  $|V(G)| < N$  do
3:    $N \leftarrow |V(G)|$ 
4:   if not  $Q, G \leftarrow \text{solve\_}K_{112}(G)$  return 0,  $G$ 
5:   if  $G$  is a triangle return 1,  $G$ 
6:   if  $\alpha > 0$  then  $G \leftarrow \text{solve\_}T_{31}(G, \alpha)$ 
7: end while
8: return  $\infty, \emptyset$ 

```

The routine implementation has two subroutines: `solve_` K_{112} and `solve_` T_{31} . The `solve_` K_{112} routine applies a vertex contraction for every K_{112} subgraph and detects wether the graph contains a K_4 until no other K_{112} is found. Pseudo-code of the routine is shown in Algorithm 2.

The `solve_` T_{31} comprises a more sophisticated test. The `solve_` T_{31} routine (cf. Algorithm 3) operates on T_{31} tadpole graphs (Fig. 1B). The key idea behind the test is that T_{31} subgraphs impose only binary constraints, i.e., either $\{x, w\}$ or $\{y, w\}$ must be contracted since $T_{31} + xw + yw$ is a K_4 (the same occurs with square graphs but

Algorithm 2 `solve_` $K_{112}(G)$

```

1: for all  $\{x, y, z, w\} = K_{112}$  do
2:    $G \leftarrow G/zw$  {contract the non-complete vertices  $\{z, w\}$ }
3:   if  $K_4 \subseteq G$  return 0,  $G$ 
4: end for
5: return 2,  $G$ 

```

Algorithm 3 $\text{solve_}T_{31}(G, \alpha)$

```

1: for all  $\{x, y, z, w\} = T_{31}$  do
2:   if not  $\text{is\_3colorable}(G/xw, \alpha - 1)$  then
3:      $G \leftarrow G/yw$  {contraction  $G/xw$  failed hence contract  $G/yw$ }
4:   return  $G$ 
5:   end if
6: end for
7: return  $G$ 

```

these are not used in the current algorithm). Thus the algorithm proceeds by contracting G/yw whenever G/xw is not 3-colorable. The $\text{solve_}T_{31}$ routine also makes a call to is_3colorable when the current recursive deep is less than the maximum recursive level α .

3.2. The coloring routine (planar graphs)

The coloring routine, planar_3COL , uses the is_3colorable algorithm and lemma 2.5 to try to obtain a legal 3-coloring.

The main procedure of the algorithm consist in reducing G to a planar triangulation by means of the addition/contraction of the planar preserving edges of G (we have assumed that the graph is planar).

A key-idea is to consider the return value $\langle \infty, \emptyset \rangle$ of the is_3colorable as that “there is no evidence that the graph is not 3-colorable” so assume it is 3-colorable. To reduce G to a planar triangulation, there are basically two equivalent options given a list of planar preserving edges $p(E)$:

- (1) The first option is to apply sequentially a vertex contraction to each element $\{u, v\}$ of $p(E)$ and verify 3-colorability of G/uv , if G/uv is not 3-colorable then revert the vertex contraction and add the edge uv to G .
- (2) The second option is just to add edges first and reverting them to a vertex contraction.

Both options are equivalent, but the first one (vertex contraction) reduces the order (number of vertices) of G and this should improve the performance of the algorithm which is better when G/uv is expected to remain 3-colorable as the current graph.

At the end, if the triangulation has all degrees even then it is 3-colorable [Heawood 1898; Jensen and Toft 1995] and finding a legal 3-coloring is immediate. Otherwise, the algorithm returns “undetermined”, meaning that α was not enough to obtain a witness for the input graph. A pseudo-code is shown in the planar_3COL routine (cf. Algorithm 4).

Finally, as a way to speedup the search for a legal 3-coloring, a greedy-trial coloring algorithm is designed (try_greedy). The key idea is to try a greedy coloring from scratch re-starting the algorithm at each triangle when no legal 3-coloring is found until all triangles have been tried. For each triangle, the algorithm selects vertices sequentially and contracts them with the first non-adjacent vertex of the triangle. Thus, if the result is a triangle then a legal 3-coloring is found. This routine is shown in Algorithm 5.

3.3. Generalizing to the non-planar case

Although planar graph 3-coloring is NP-complete, reducing a non-planar graph to a planar one preserving 3-colorability usually produce a planar graph two orders of magnitude bigger than the original one and this may signify that for practical applications, the resulting planar graphs could be intractable from a practical application perspec-

Algorithm 4 planar_3COL(G, α)

```

1:  $Q, H \leftarrow \text{is\_3colorable}(H, \alpha)$ 
2: if  $Q \neq \infty$  return  $Q, H$ 
3: else if  $\text{try\_greedy}(H)$  return 1,  $H$ 
4: while  $G$  is not a planar triangulation do
5:    $u, v \leftarrow$  a valid planar preserving edge
6:    $H \leftarrow G/uv$ 
7:    $Q, H \leftarrow \text{is\_3colorable}(H, \alpha)$ 
8:   if  $Q = 1$  return 1,  $H$ 
9:   else if  $Q = 0$  then  $G \leftarrow G + uv$ 
10:  else if  $\text{try\_greedy}(H)$  return 1,  $H$ 
11:  else  $G \leftarrow H$  {undetermined case, assume 3-colorable}
12: end while
13: if triangulation  $G$  has an odd vertex return  $\infty$ 
14: return 1,  $G$ 

```

Algorithm 5 try_greedy(G)

```

1: for all triangles  $\{x, y, z\} \in G$  do
2:    $H \leftarrow G$ 
3:   for all  $w \in V(G) \setminus \{x, y, z\}$  do
4:     if  $\{x, w\} \notin E(G)$  then  $H \leftarrow H/xw$ 
5:     else if  $\{y, w\} \notin E(G)$  then  $H \leftarrow H/yw$ 
6:     else if  $\{z, w\} \notin E(G)$  then  $H \leftarrow H/zw$ 
7:     else break
8:   end for
9:   if  $G$  is a triangle return 1,  $H$ 
10: end for
11: return 0,  $G$ 

```

tive. Hence, it looks imperative to find the way to generalize the algorithm to operate directly on non-planar graphs.

A natural way of generalizing the algorithm, without finding new structural insights or heuristics, is to increase the α level up to an “acceptable” number of errors in the decision algorithm. Also this is the right way of approaching the problem from the parametric complexity methodology, i.e., to determine a fixed value for the parameter (α) for which the problem is tractable.

However, to provide an algorithm that could be proved to be valid in the general case, we should take care of triangle-free graphs, so a routine based on tadpole processing seems to be incomplete. The most simple modification is to replace the tadpole test by a test based on the edge-addition/vertex-contraction of the end-vertices of a P_2 path x, y, z :

- (1) Select a P_2 , add edge xz to G and test colorability for $G + xz$ and α ;
- (2) If $G + xz$ is not 3-colorable then xz is a forced vertex contraction;
- (3) Contract xz and return the graph G/xz to continue working with it.

It is very easy to verify that this algorithm is correct since a graph without an induced P_2 subgraph is a complete graph. A pseudo-code is shown in Algorithm 6.

As for the planar version, the general 3-coloring algorithm relies in the `is_3colorable` routine for supporting the search for a legal 3-coloring of the input graph G . However, the key-idea in this case is to build a complete vertex, i.e., a vertex

Algorithm 6 solve_non_edge(G, α)

```

1: for all  $\{x, y, z\} = P_2$  do
2:   if not is_3colorable( $G + xz, \alpha - 1$ ) then
3:      $G \leftarrow G/xz$  {edge-addition  $G + xz$  failed hence contract  $G/xz$ }
4:   return  $G$ 
5:   end if
6: end for
7: return  $G$ 

```

Algorithm 7 general_3COL(G, α)

```

1:  $Q, H \leftarrow \text{is\_3colorable}(H, \alpha)$ 
2: if  $Q \neq \infty$  return  $Q, H$ 
3: else if try_greedy( $H$ ) return 1,  $H$ 
4:  $u \leftarrow$  the highest degree vertex of  $G$ 
5: while  $x$  is a non-complete vertex, i.e.,  $N(x) \subset V(G) - \{x\}$  do
6:    $v \leftarrow$  a non-neighbor of  $x$  maximizing  $|N(x) \cup N(y)|$ 
7:    $H \leftarrow G/uv$ 
8:    $Q, H \leftarrow \text{is\_3colorable}(H, \alpha)$ 
9:   if  $Q = 1$  return 1,  $H$ 
10:  else if  $Q = 0$  then  $G \leftarrow G + uv$ 
11:  else if try_greedy( $H$ ) return 1,  $H$ 
12:  else  $G \leftarrow H$  {undetermined case, assume 3-colorable}
13:   $u \leftarrow$  the highest degree vertex of  $G$ 
14: end while
15: if  $N(x)$  is bipartite return 1, a three-coloring of  $G$ 
16: return  $\infty, G$ 

```

Algorithm 8 BFS_3COL(G)

```

1: for  $\alpha = 0$  to  $\infty$  do
2:    $Q, G = \text{general\_3COL}(G, \alpha)$ 
3:   if  $Q \in \{0, 1\}$  return  $Q, G, \alpha$ 
4: end for

```

joined to all the remaining vertices of the graph, so that for testing 3-colorability at the end it suffices to test 2-colorability of its neighborhood. A pseudo-code of the general 3-coloring algorithm (general_3COL) is shown in Algorithm 7.

Finally, an automated algorithm can be developed to eliminate the need of specifying the α parameter. Even more, this last routine (Algorithm 8) computes the function $\alpha(G)$ which is of theoretical interest.

4. ANALYSIS OF THE ALGORITHM

The average-case complexity, worst-case complexity and experimental performance of the algorithm are analyzed. The average-case is informally presented as a mean of establishing the expected behavior over different kind of instances. The worst-case analysis establishes the order (Big O) of the algorithm. Finally, the experimental analyses confront the algorithm with a series of graph distributions to study its performance and contrast it with the theoretical results.

Table I: Description of the a priori expected performance of the algorithm with respect to 3-colorability and graph density parameters

KIND	3-COLORABLE	NOT 3-COLORABLE
SPARSE ($d < d^*$)	High probability of a short running-time due to the existence of many legal colorings.	High probability of a short running-time since almost all non-3-colorable graphs contain a K_4 and hence the probability of obtaining a K_4 -free non-3-colorable graph decreases very fast when the average degree falls below the phase transition threshold.
d^*	Harder instances.	Harder instances.
DENSE ($d > d^*$)	High probability of a short running-time due the existence of many K_{112} subgraphs that prune the search, e.g., graphs tend to be uniquely colorable.	High probability of a short running time due the existence of many small 3-uncolorability witnesses due to the average degree, e.g. too many K_4 -subgraphs.

*Phase transition threshold: average degree $d \simeq 4.69$.

4.1. Average-case (expected) complexity

Table I shows the average case (expected) performance of the algorithm with respect to the type of the instance (Yes or No) and the density of the graph, i.e., above/below the phase transition threshold.

In all the cases (except at the phase transition threshold) there is a high probability of a short running time. A priori, it may look that the worst-case should occur on the sparse non-3-colorable graphs. This observation comes from the fact that for this class of graphs it is more complex to obtain a K_4 by random edge additions and vertex contractions, nevertheless, some restrictions apply! Since the proportion of non-3-colorable graphs decreases fast below the threshold and almost all non-colorable graphs contain a K_4 then the probability of obtaining a K_4 -free non-3colorable graph below the threshold is very small. Even more, it is known that vertex 3-colorability of a graph with maximum vertex degree 3 can be determined in polynomial time [Steinberg 1993]. Also every vertex of max degree 2 can be removed from the graph without affecting 3-colorability so non-3-colorable sparse graphs are very rare below $c \lesssim 4$ (this follows from the sharp-thresholds theory).

Hence in almost all cases a short running time is expected. By short I mean significantly shorter than the worst-case exact upper bound.

4.2. Worst-case complexity

To determine the computational complexity (g) of the whole algorithm we will start by analyzing from the central routine: `is3colorable`. The routine admits a special parameter α that control the level of recursive calls. In order to analyze its complexity, the recursion is fixed to $\alpha = 0$ and once obtained the complexity for $\alpha = 0$ the complexity for $\alpha > 0$ is established.

The `is3colorable` routine depends on the complexity of two subroutines: `solve $_{K_{112}}(G)$` and `solve $_{T_{31}}(G, \alpha)$` , which are called inside a while loop. However, for $\alpha = 0$ the `solve $_{T_{31}}(G, \alpha)$` routine is not called.

The routine `solve $_{K_{112}}(G)$` has complexity of order at most $O(n^4)$ since it explores each K_{112} whose number grows as the number of combinations of four elements in the vertices of G . However, a deeper analysis show us that at each iteration of the while

loop, the algorithm performs a vertex contraction until there is no other K_{112} . This means that the loop is bounded by the number of edges of the complement of G which has a quadratic $O(n^2)$ order in the number of vertices. Hence:

$$g(\text{solve_}K_{112}) = \binom{n}{2} \quad (15)$$

$$= O(n^2) \quad (16)$$

The complexity of `is3colorable` routine also depends on a while loop that calls sequentially the `solve_` $K_{112}(G)$ routine. The while loop will iterate until the graph becomes a triangle $|V(G)| = 3$ or earlier, when no other K_{112} subgraph exists and so no vertex contraction is performed. Then, there are only three possibilities at each iteration of the while loop:

- (1) Decrease the number of elements of V .
- (2) End due to the number of elements in V has not decreased.
- (3) End due to G becomes a triangle, i.e., $|V| = 3$.

The worst-case scenario is the third case; when the loop ends due to $|V| = 3$, since this imply that the loop has been executed *exactly* $n - 3$ times (which is the maximum possible). Thus the while loop has linear complexity $O(n)$ and hence the whole `is_3colorable` routine for $\alpha = 0$ has complexity:

$$\alpha = 0; \quad (17)$$

$$g(\text{is_3colorable}) = n \cdot (g(\text{solve_}K_{112})); \quad (18)$$

$$g(\text{is_3colorable}) = O(n) \cdot (O(n^2)); \quad (19)$$

$$g(\text{is_3colorable}) = O(n^3) \quad (20)$$

Now, for the $\alpha > 0$ case, first consider $\alpha = 1$. In this case, the routine `solve_` T_{31} is called. This routine has complexity $O(n^3)$ since it explores each T_{31} tadpole subgraph (whose number grows as the number of combinations of four elements) but it works by contracting either $\{x, w\}$ or $\{y, w\}$ so it is only needed to look for unique x, y, w combinations of the vertices of tadpole subgraphs.

Since the routine recursively calls the `is_3colorable` algorithm with $\alpha - 1$ recursion-level in each iteration, the worst-case complexity of `solve_` T_{31} is:

$$\alpha = 1 \quad (21)$$

$$g(\text{solve_}T_{31}) = \binom{n}{3} \cdot O(n^3) \quad (22)$$

$$g(\text{solve_}T_{31}) = O(n^3) \cdot O(n^3) \quad (23)$$

$$g(\text{solve_}T_{31}) = O(n^6), \quad (24)$$

and thus:

$$\alpha = 1 \quad (25)$$

$$g(\text{is_3colorable}) = O(n^3) \cdot O(n^6) \quad (26)$$

$$= O(n^9), \quad (27)$$

and therefore, for a constant value: $\alpha = k$, the complexity of the decision algorithm is:

$$\alpha = k \quad (28)$$

$$g(\text{is_3colorable}) = O(n^3).O(n^6)^k \quad (29)$$

Now, by lemma 2.5, the complexity of an algorithm that finds a 3-coloring is just one order higher:

$$g(\text{_3COL}) = O(n).O(n^3).O(n^6)^k \quad (30)$$

4.3. Experimental work

The algorithm has been thoroughly evaluated through experiments over significant samples taken from three different graph distributions:

- (1) Pseudo-random planar graphs.
- (2) Random 4-regular planar graphs.
- (3) Erdős-Rényi (connected) random graphs.

Random planar graphs [Denise et al. 1996; Bodirsky et al. 2003] are more complex and its definition and sampling methods are more difficult to implement. Instead, we opted for a simpler approach to generate “pseudo random planar graphs”. The procedure is to generate a maximal planar graph and select edges from it uniformly at random (i.e. with equal probability) to create another graph called a pseudo random graph. For this purpose, we used the “Create Random Planar Graph” algorithm implementation used by Hochstättler and Schliep [2010] (Gato - the Graph Animation Toolbox⁵) for the creation of such random planar graphs. Only one modification was included to avoid the obtention of too much graphs containing a K_4 subgraph. The idea consist in trying to generate a K_4 -free planar graph during 100 attempts returning the first encountered K_4 -free graph, otherwise return the 100th generated graph.

Random 4-regular planar graphs are also complex to generate. Here, the procedures described by Manca [1979], Lehel [1981] and Broersma et al. [1993] to generate all the 4-regular planar graphs have been used. In particular, theorem 2 of Broersma et al. [1993] is used to generate 4-regular planar graphs. However, there is no theory defining a random 4-regular planar graph, so an ad hoc distribution has been specified trying to balance the proportion of yes/no instances. The distribution has been obtained assigning a probability to each graph transformation (see Broersma et al. 1993): $P(\bar{\phi}_A) = .80$, $P(\bar{\phi}_B) = .05$, $P(\bar{\phi}_C) = .10$ and $P(\bar{\phi}_F) = .05$.

The Erdős and Rényi [1960] random graph, is a very well-known model that allows sampling from graphs uniformly at random and also for a fixed number of vertices or edges. We followed standard methods to sample from this distribution. The only modification is that the generation of a connected graph is assured by generating first a simple path passing through all the vertices and then adding the remaining random edges.

The experiments are designed to study the behavior (not just the performance or running-time) of the proposed algorithm. For this purpose, there are curves evaluating its performance over a particular graph distribution and also there is an initial comparative plot against the backtracking algorithm. The use of backtracking is restricted to the study of the algorithm’s scalability since it is not possible to use backtracking consistently beyond the 100 vertex barrier due to its exponential growth.

⁵CATBox: <http://schliep.org/CATBox>, Gato: <http://gato.sf.net>.

Table II: Sample used in the scalability test.

Sample property	Value
Sample type:	Random planar graphs.
Sample size:	9000 graphs.
Sample range:	From 10 to 100 vertices, counting by 1.
Group size:	100 graphs per vertex number.

All experiments were developed in the Python⁶ programming language using its standard libraries and other libraries developed by the current author. Other software includes the planarity library⁷ [Boyer and Myrvold 2004] as well as the above mentioned Gato (the Graph Animation Toolbox) libraries. The experiments were realized on common personal computers and no parallelism was used. All time measurements are done in seconds using the python's `time.clock()`⁸ function.

4.3.1. Scaling factor compared against backtracking. The first part of the experimental analysis is a comparison between the scaling factor of the proposed algorithm against simple backtracking, in order to determine the differences in the behavior of both algorithms.

This experiment consisted in generating uniformly random planar graph instances from 10 to 100 vertices (counting by 1 and generating 100 graphs for each number) and solving each instance with both algorithms. Table II shows the parameters of the sample used in the experiment.

For each algorithm, the mean and maximum (max) running-times were recorded as well as some other relevant statistics. Times labeled as t_1 correspond to the backtracking algorithm while t_2 times correspond to the proposed parametric algorithm. Comparative plots are shown in Fig. 2 where there are six plots from (a) to (f).

Fig. 2a shows the running-times as a function of the number of vertices for both kinds of instance types and for both algorithms. Fig. 2b shows the proportion of 3-colorable and non-3-colorable graphs over the total number of graphs per number of vertices. It can be seen that the distribution tends to be uniform. Figures 2c and 2d also show the running-times as a function of the number of vertices but discriminating by the instance types (yes-or-no) so that subtle differences can be observed.

The general results indicate that there is a crossing point in which backtracking continues to grow exponentially while the proposed algorithm remains polynomial (cf. Fig. 2a at around 50 vertices), so a clear difference in the behavior of both algorithms is observed. This difference is clearer in the non-3-colorable instances (cf. Fig. 2c) where the maximum running-times of the parametric algorithm are lower in all the cases. Nevertheless, for the 3-colorable instances (cf. Fig. 2d) the difference starts to be clear around graphs on 50 vertices.

Moreover, when running-times are compared as a function of the average degree, there is also a significant difference in the behavior of both algorithms. For non-3-colorable instances the parametric algorithm exhibits an almost constant performance (cf. Fig. 2e) and also a totally uncorrelated curve against backtracking, which on the contrary is very sensitive to the average degree. This difference, although in a slightly minor degree, can be also observed in the 3-colorable instances as shown in Fig. 2f.

⁶<http://www.python.org/>

⁷<http://code.google.com/p/planarity>

⁸See <http://docs.python.org/library/time.html>.

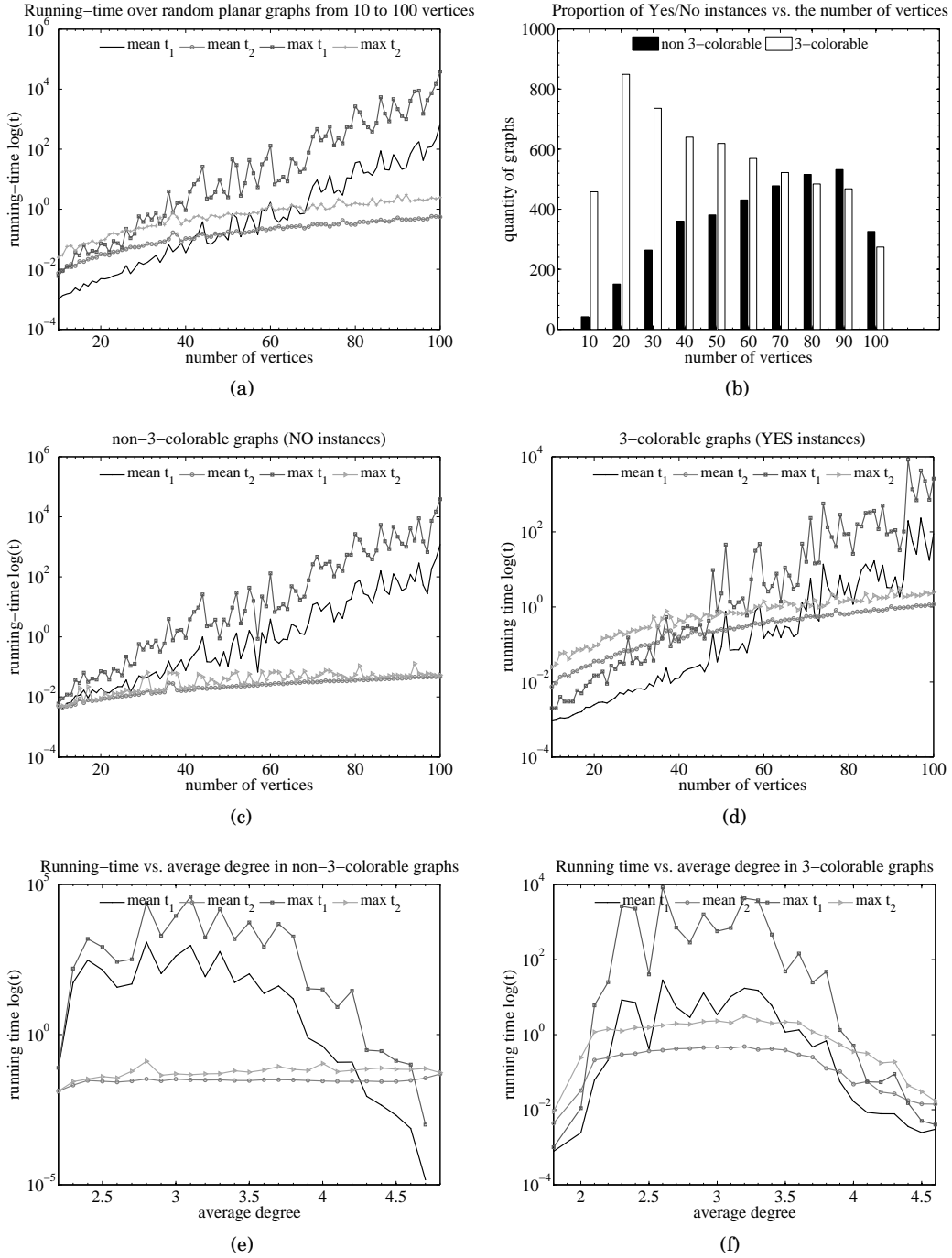


Fig. 2: Runtime analysis of the backtracking (t_1) vs. the proposed parametric algorithm (t_2) over random planar graphs between 10 and 100 vertices.

Table III: Sample used in the random planar graphs test.

Sample property	Value
Sample type:	Random planar graphs.
Sample size:	10000 graphs.
Sample range:	From 100 to 1000 vertices, counting by 100.
Group size:	100 graphs per vertex group.

4.3.2. Pseudo-random planar graphs. This experiment consisted in generating uniformly random planar graph instances from 100 to 1000 vertices (counting by 100 and generating 1000 graphs for each number) and solving each instance with the parametric algorithm. Table III shows the parameters of the sample used in the experiment. It is not possible to compare the results against backtracking due to its exponential-grow running-time.

For each instance type (Yes/No), the mean and maximum (max) running-times where recorded, as well as some other relevant statistics. Comparative plots are shown in Figures 3 and 4.

Fig. 3a shows the running-times as a function of the number of vertices for both kinds of instance types. Fig. 3b shows the proportion of 3-colorable and non-3-colorable graphs over the total number of graphs per number of vertices. It can be seen that the distribution is far from being uniform. Figures 4a and 4b also show the running-times as a function of the number of vertices but discriminating by the average degree.

The results indicate that there is a very significant difference in running-times depending on the instance type (cf. Fig. 3a). This difference is expected since the proposed algorithm returns earlier when a 3-uncolorability witness is found. Even in the case when the average degree is considered the difference is high (cf. Figures 4a and 4b).

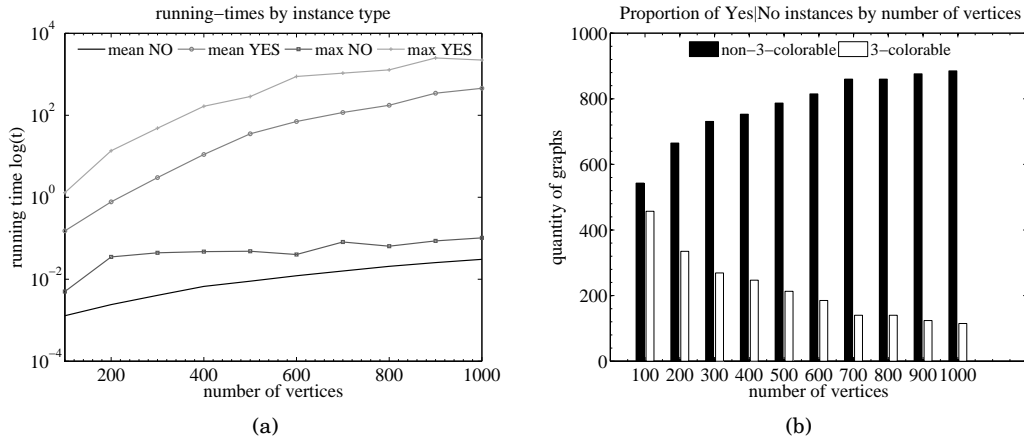


Fig. 3: Runtime analysis over random planar graphs between 100 and 1000 vertices (counting by 100 and generating 1000 graphs for each number).

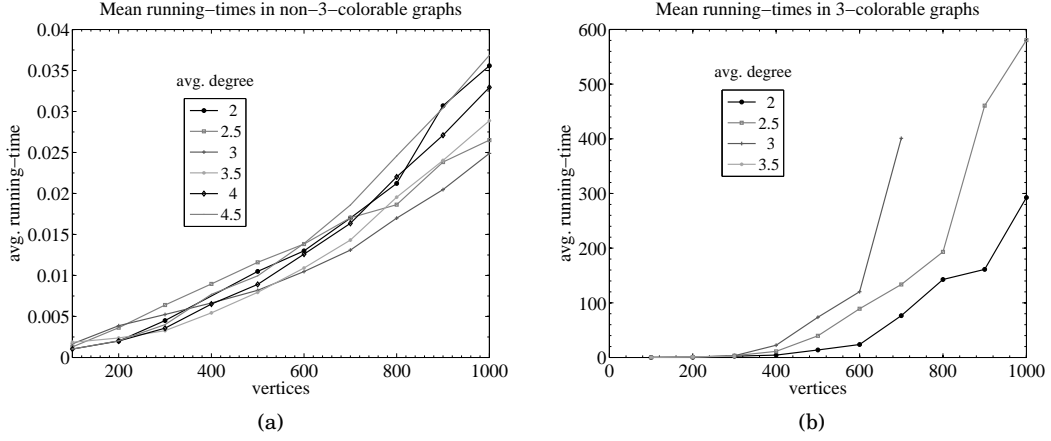


Fig. 4: Runtime analysis over random planar graphs considering instance-type and average degree.

Table IV: Sample used in the random planar 4-regular graphs test.

Sample property	Value
Sample type:	Random planar 4-regular graphs.
Sample distribution ^a :	$P(\bar{\phi}_A) = .80$, $P(\bar{\phi}_B) = .05$, $P(\bar{\phi}_C) = .10$ and $P(\bar{\phi}_F) = .05$
Sample size:	10000 graphs.
Sample range:	From 100 to 1000 vertices, counting by 100.
Group size:	1000 graphs per vertex group.

^asee Broersma et al. [1993] for the exact meaning of each graph transformation operation, i.e., $\bar{\phi}_A$, $\bar{\phi}_B$, $\bar{\phi}_C$ and $\bar{\phi}_F$.

4.3.3. Random planar 4-regular graphs. In this experiment, graphs are sampled from an ad hoc distribution over the 4-regular planar graphs. The sample consisted in generating graph instances from 100 to 1000 vertices (counting by 100 and generating 1000 graphs for each number) and solving each instance with the parametric algorithm. Table IV shows the parameters of the sample used in the experiment.

For each instance type (Yes/No), the mean and maximum (max) running-times where recorded, as well as some other relevant statistics. Comparative plots are shown in Figures 3 and 4.

Fig. 3a shows the running-times as a function of the number of vertices for both kinds of instance types. Fig. 3b shows the proportion of 3-colorable and non-3-colorable graphs over the total number of graphs per number of vertices. It can be seen that the distribution is far from being uniform. Figures 4a and 4b also show the running-times as a function of the number of vertices but discriminating by the average degree.

The results indicate that there is a very significant difference in running-times depending on the instance type (cf. Fig. 3a). This difference was expected since the proposed algorithm returns earlier when a 3-uncolorability witness is found. Even in the case when the average degree is considered, the observed difference is very high (cf. Figures 4a and 4b).

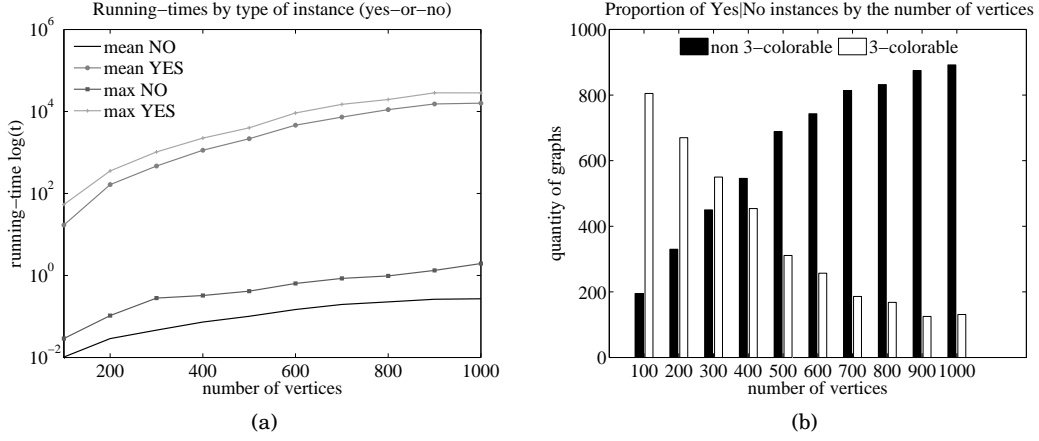


Fig. 5: Runtime analysis over random 4-regular planar graphs between 100 and 1000 vertices

Table V: Sample used in the random graphs test.

Sample property	Value
Sample type:	Erdős-Rényi random graphs.
Sample size:	10000 graphs.
Sample range:	100 vertices.

4.3.4. Erdős-Rényi random graphs. In the last experiment, graphs are sampled from the well-known Erdős-Rényi random graphs distribution. The sample consisted in generating graph instances of 100 vertices, generating in total 10000 graphs, and solving each instance with the parametric algorithm. Table V shows the parameters of the sample used in the experiment. For each instance type (Yes/No), the mean and maximum (max) running-times were recorded, as well as some other relevant statistics. Comparative plots are shown in Fig. 6.

Fig. 6a shows the quantity of 3-colorable and non-3-colorable graphs as a function of the average degree, i.e., a phase transition plot. In this case occurring at around $d = 4.74$. Fig. 6b shows the quantity of graphs corresponding to each $\alpha(G)$ value. As predicted by the theory, almost all graphs have $\alpha(G) \leq k$ (for some integer k) and the proportion of graphs decreases exponentially with α below the line of $1/2^{\alpha+1}$. These results confirm the established theoretical bounds.

Figures 6c and 6d show the running-time as a function of the average degree. It can be observed that in the random graphs case, the difference in running-time is not as high as the difference observed in the planar graphs case.

Also, there is a difference in the location of the harder instances for each kind of instance-type: the harder instances for the non-3-colorable case are located around an average degree of $d \approx 5$ while for 3-coloring instances are located slightly below an average degree of $d \approx 4.8$. Although the numbers seems to be very close the shape of the running-time curves are not. The shape of the running-time curve in 6d falls sharply after 5 while the shape of the running-time curve in 6c does not. This may indicate that there is a true difference in the location of the harder instances depending on its class (yes-or-no).

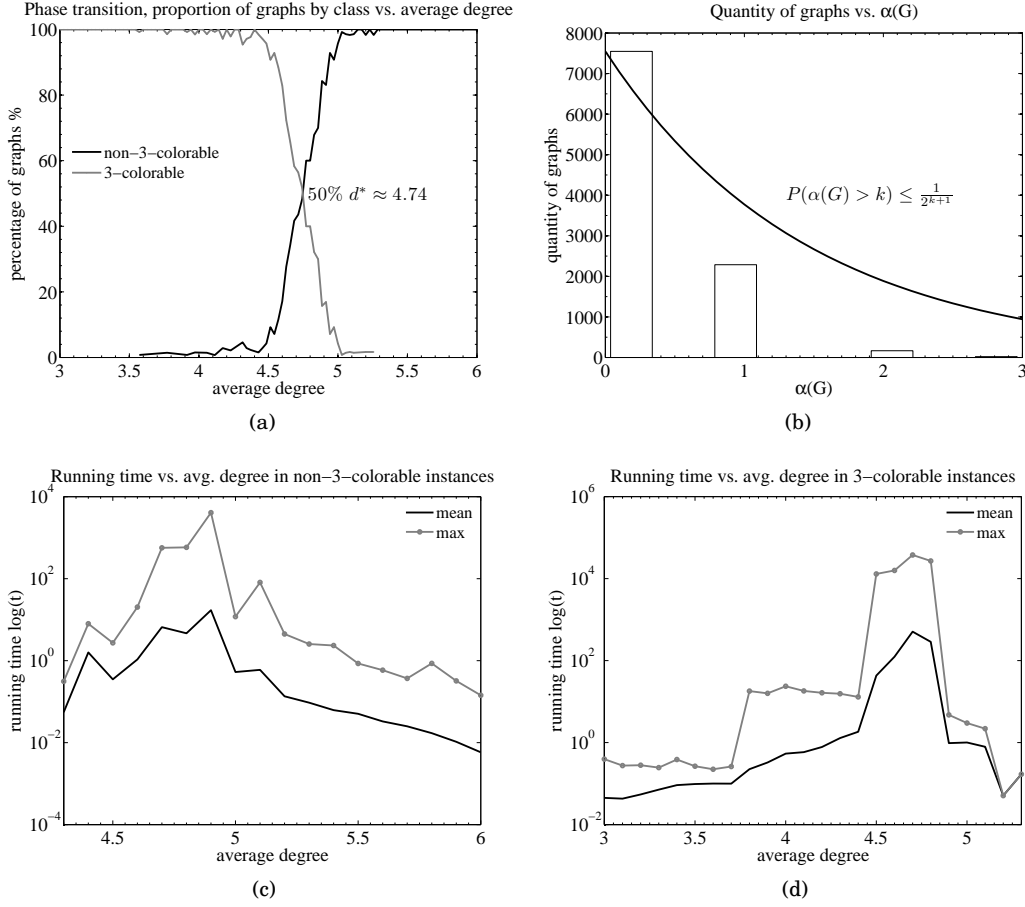


Fig. 6: Runtime analysis of the algorithm for random graphs

5. CONCLUSIONS AND FINAL DISCUSSION

In this article, a parametric exact 3-coloring algorithm has been presented. This is (to the best of my knowledge) the first algorithm of its kind for the 3-coloring problem.

The maximal complexity of the algorithm is controlled by the parameter (α) that bounds the recursion deep and determines its running-time: $O(n^{f(\alpha)})$. The algorithm relies on the efficient search of 3-uncolorability witnesses. Here, a formal definition of 3-uncolorability witness has been introduced. This is the central theoretical concept that allowed the development of the proposed algorithm. The definition of 3-uncolorability witness presented here is (to the best of my knowledge) the first one that is formally defined and the most naturally related to the 3-coloring problem. Also, a close relation between the Zykov [1952] theorem and the definition of 3-uncolorability witnesses defined here was shown.

A very significant feature of 3-uncolorability witnesses is that it is possible to obtain them from small subgraphs of a particular graph, indeed, as small as four vertices (i.e. by finding a K_4 subgraph). Hence, an interesting theoretical analysis that should follow is to study the behavior of $\alpha(G)$ on 4-critical graphs since in that class there is no subgraph with chromatic number four and hence finding forced vertex contractions

should be harder (see Mizuno and Nishihara 2008 for a good initial development of this idea). Hence, a classification of 4-critical graphs based on $\alpha(G)$ could lead to very significant results.

There is an interesting symmetry between colorings and uncolorability witnesses:

- In order to show that a graph is 3-colorable it suffices to encounter just one legal coloring, nevertheless a legal coloring must assign a color to all the vertices of the graph without violating any constraint since it remains hard to determine if a partial coloring is extensible to all the vertices of the graph.
- Instead, in order to show that a graph is not 3-colorable it is needed to verify that none of all possible 3-colorings is a legal one, nevertheless for finding a 3-uncolorability witness it suffices to encounter just one non-3-colorable subgraph (e.g. a 4-critical subgraph), i.e., a small graph.

Thus, while for huge graphs just verifying a legal 3-coloring could become impossible in practice, it remains practical (at least in theory!) to determine 3-uncolorability even for huge graphs.

Hence, in principle, finding uncolorability witnesses can be assumed to be at least of the same kind than finding colorings. Thus, there shouldn't be any problem in the development of 3-coloring algorithms based on searching for 3-uncolorability witnesses, that eventually reach the same level of sophistication and performance than its colorings-based peers.

Also if the algorithm is used as a heuristic, for example to test whether a solution can be found quickly ("just by chance") with a lower (efficient) α , the algorithm will search for both 3-colorings and 3-uncolorability witnesses at the same time. This makes a clear difference to the use of backtracking, greedy-based or randomized 3-coloring algorithms. Besides, this feature is especially important since then it is not necessary to trust the correctness of the algorithm itself, nor the particular implementation used, to recognize that the provided solution is correct since the verification of the result can be done efficiently using just only the solution provided (a legal 3-coloring or a 3-uncolorability witness).

The developed theoretical analysis guarantees some good features of the proposed algorithm. The most important one, both for practical and theoretical purposes is that while the algorithm relies on the value of α to be able to find a witness, the probability that $\alpha(G) > k$ decreases at the rate $P(\alpha(G) > k) \leq 1/2^{k+1}$, e.g., for $k = 19$ there is less chance than one in a million of not obtaining a solution with the proposed polynomial algorithm (i.e. 0.999999 of probability of success), assuming that the input is a random graph.

Thus, while certainly beyond some value of α the running-times would become prohibitive given the current state of the computing machinery, the developed algorithm scales polynomially and the probability of obtaining a solution (success) grows exponentially on α , i.e.:

$$O\left(n^{f(\alpha=k)}\right) = O(n).O(n^3).O(n^6)^k;$$

$$P(\alpha(G) > k) \leq 1/2^{k+1},$$

hence any step (i.e. any investment) in computing power technology will mean a huge (exponential) growing step in the class of tractable 3-coloring instances. Perhaps it could be the case that we can achieve at least a "technological tractability"? I mean a guaranteed number of instances such that almost all computational problems of practical interest could be solved for $\alpha(G) \in [0, k]$, for some integer k .

It should be also observed that increasing α as the result of technological progress implies that $\alpha \neq f(\text{input})$. Does technological progress imply a polynomial algorithm for 3-colorability?

On the other hand, corollary 1.3 (10) establishes a new definition of computational complexity classes under the established parametric conditions: $\text{NP}(\alpha)$ and $\text{CoNP}(\alpha)$ are respectively the subsets of problems which are in NP or CoNP that given a (maximum) value of α are also in P. This, as proved here, implies that as $\alpha \rightarrow \infty$ $(\text{NP} \cup \text{CoNP}) \setminus \text{P} \rightarrow \emptyset$, which could have some non-trivial interpretations, for instance, following the “physical process criterion for mathematical truth” [Aaronson 2003]: “we should expect a mathematical question to have a definite answer, if and only if we can phrase the question in terms of a physical process we can imagine”⁹.

Would any physical constraint to $\alpha(G)$ (i.e. an upper-bound) imply $\text{P} = \text{NP}$ since $\text{P} \neq \text{NP}$ implies $\alpha(G) \in [0, \infty)$?

Thus, the following questions naturally appear:

- (1) Is there any graph construction mechanism that allows the generation of graphs with arbitrarily large $\alpha(G)$? And if so, what is the complexity of such algorithm?
- (2) Even more, there is any possible physical machine able to generate one of such arbitrarily-large- $\alpha(G)$ -graph instances in finite time?
- (3) In other words, can we assume that it is possible to encounter or to generate, in finite time, graphs with arbitrarily large $\alpha(G)$ without violating any physical law?

In addition, since 3-colorability is NP-complete and to each graph corresponds a unique $\alpha(G)$ then a classification based on $\alpha(G)$ of all the NP-complete problem instances can be done by a reduction of each problem instance to a 3-coloring instance G such that $\alpha(G) = k$ for some $k \in \mathbb{N}$.

However, can we define NP as follows?

$$\text{NP} = \bigcup_{\alpha=0}^{\infty} \text{NP}(\alpha), \quad (31)$$

i.e., can then NP be defined as the infinite union of problems in P?

Finally, even determining the infiniteness of $\alpha(G)$, is there –as with maximal degree four ($\Delta(G) \leq 4$), a $k \in \mathbb{N}$ such that determining 3-colorability over a class of graphs with $\alpha(G) \leq k$ is still NP-complete?, i.e., $\text{P} = \text{NP}$?

In the maximal degree case, we know that 3-colorability restricted to $\Delta(G) \leq 4$ is still NP-complete. Nevertheless, the problem is to determine if a polynomial algorithm exists or not.

On the contrary, in the finite $\alpha(G)$ case, we know that 3-colorability restricted to $\alpha(G) \leq k$ is in P. Nevertheless, the problem is to determine whether it is NP-complete for some $k \in \mathbb{N}$.

REFERENCES

- AARONSON, S. 2003. Is P versus NP formally independent? *Bulletin of the EATCS* 81, 109–136.
- ALON, N., BALOGH, J., BOLLOBÁS, B., AND MORRIS, R. 2010. The structure of almost all graphs in a hereditary property. *Journal of Combinatorial Theory, Series B*.
- APPEL, K., HAKEN, W., AND KOCH, J. 1977a. Every planar map is four colorable. Part I: Discharging. *Illinois Journal of Mathematics* 21, 3, 429–490.
- APPEL, K., HAKEN, W., AND KOCH, J. 1977b. Every planar map is four colorable. Part II: Reducibility. *Illinois Journal of Mathematics* 21, 3, 491–567.

⁹“most computer scientists would agree that the *extended* (or polynomial-time) Church-Turing thesis depends on physical laws, and indeed is probably falsified by quantum mechanics” [Aaronson 2003].

- ARORA, S. AND BARAK, B. 2009. *Computational Complexity: A Modern Approach* 1st Ed. Cambridge University Press, New York, NY, USA.
- ARORA, S. AND CHLAMTAC, E. 2006. New approximation guarantee for chromatic number. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*. STOC '06. ACM, New York, NY, USA, 215–224.
- BERGER, B. AND ROMPEL, J. 1990. A better performance guarantee for approximate graph coloring. *Algorithmica* 5, 1, 459–466.
- BES, J.-N. AND JEGOU, P. 2005. Proving graph un-colorability with a consistency check of CSP. In *Tools with Artificial Intelligence, 2005. ICTAI 05. 17th IEEE International Conference on*. 2 pp. –694.
- BLUM, A. 1994. New approximation algorithms for graph coloring. *Journal of the ACM (JACM)* 41, 3, 516.
- BODIRSKY, M., GRÖPL, C., AND KANG, M. 2003. Generating labeled planar graphs uniformly at random. *Automata, Languages and Programming*, 191–191.
- BOETTCHER, S. AND PERCUS, A. G. 2004. Extremal optimization at the phase transition of the three-coloring problem. *Physical Review E* 69, 6, 066703.
- BOPPANA, R. B., HASTAD, J., AND ZACHOS, S. 1987. Does co-NP have short interactive proofs? *Information Processing Letters* 25, 2, 127–132.
- BORODIN, O. V. 1996. Structural properties of plane graphs without adjacent triangles and an application to 3-colorings. *Journal of Graph Theory* 21, 183–186.
- BORODIN, O. V., GLEBOV, A., RASPAUD, A., AND SALAVATIPOUR, M. 2005. Planar graphs without cycles of length from 4 to 7 are 3-colorable. *Journal of Combinatorial Theory, Series B* 93, 2, 303 – 311.
- BORODIN, O. V., GLEBOV, A. N., MONTASSIER, M., AND RASPAUD, A. 2009. Planar graphs without 5- and 7-cycles and without adjacent triangles are 3-colorable. *Journal of Combinatorial Theory, Series B*. 99, 668–673.
- BOYER, J. AND MYRVOLD, W. 2004. On the cutting edge: Simplified $O(n)$ planarity by edge addition. *Journal of Graph Algorithms and Applications* 8, 3, 241–273.
- BROERSMA, H., DUIJVESTIJN, A., AND GÖBEL, F. 1993. Generating all 3-connected 4-regular planar graphs from the octahedron graph. *Journal of graph theory* 17, 5, 613–620.
- CHARTRAND, G. AND ZHANG, P. 2008. *Chromatic Graph Theory* 1st Ed. Chapman & Hall/CRC.
- COOK, S. A. 1971. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*. STOC '71. ACM, New York, NY, USA, 151–158.
- CORNEIL, D. G. AND GRAHAM, B. 1973. An algorithm for determining the chromatic number of a graph. *SIAM Journal of Computing* 2, 4, 311–318.
- CULBERSON, J. AND GENT, I. 2001. Frozen development in graph coloring. *Theoretical computer science* 265, 227–264.
- DAILEY, D. P. 1980. Uniqueness of colorability and colorability of planar 4-regular graphs are np-complete. *Discrete Mathematics* 30, 3, 289 – 293.
- DENISE, A., VASCONCELLOS, M., AND WELSH, D. 1996. The random planar graph. *Congressus numerantium*, 61–80.
- DOWNNEY, R. AND FELLOWS, M. 1999. *Parameterized complexity*. Vol. 5. Springer New York.
- ERDŐS, P. AND RÉNYI, A. 1960. On the evolution of random graphs. *Publications of the Matkematical Institute of the Hungarian Academy of Sciences* 5.
- ERDOS, P., KLEITMAN, D., AND ROTHSCCHILD, B. 1976. Asymptotic enumeration of k -free graphs. In *Colloquio Internazionale sulle Teorie Combinatorie (Rome, 1973)*. Vol. 2. Atti dei Convegni Lincei, 17, Accad. Naz. Lincei, Roma, 19–27.
- FLUM, J. AND GROHE, M. 2006. *Parameterized complexity theory (texts in theoretical computer science. an eatcs series)*.
- FORTNOW, L. AND SIPSER, M. 1988. Are there interactive proofs for co-NP languages? *Information Processing Letters* 28, 249–251.
- GAREY, M. R. AND JOHNSON, D. S. 1976. The complexity of near-optimal graph coloring. *Journal of the ACM (JACM)* 23, 43–49.
- GAREY, M. R. AND JOHNSON, D. S. 1979. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., San Francisco.
- GOLDREICH, O. 2008. *Computational complexity: a conceptual perspective*. Cambridge University Press.
- HALLDÓRSSON, M. 1993. A still better performance guarantee for approximate graph coloring. *Information Processing Letters* 45, 1, 19–23.
- HEAWOOD, P. 1898. On the four-colour map theorem. *Quarterly Journal of Pure and Applied Mathematics* 29, 270–285.

- HOCHSTÄTTLER, W. AND SCHLIEP, A. 2010. *CATBox: An Interactive Course in Combinatorial Optimization* 1st, Softcover Ed. Springer.
- HOGG, T., HUBERMAN, B., AND WILLIAMS, C. 1996. Phase transitions and the search problem. *Artificial intelligence* 81, 1-2, 1–15.
- JENSEN, T. R. AND TOFT, B. 1995. *Graph coloring problems*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, Chichester-New York-Brisbane-Toronto-Singapore.
- JOHNSON, D. S. 1974a. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences* 9, 3, 256 – 278.
- JOHNSON, D. S. 1974b. Worst case behavior of graph coloring algorithms. In *Proceedings of the Fifth South-eastern Conference on Combinatorics, Graph Theory and Computing (Florida Atlantic Univ., Boca Raton, Fla., 1974, F. Hoffman et al., eds.)*. 513–527.
- JOHNSON, D. S. 2002. A theoreticians guide to the experimental analysis of algorithms. In *Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges*. American Mathematical Society, 215–250.
- JONES, M. 2008. *Artificial Intelligence: A Systems Approach*. Computer Science. Jones & Bartlett Publishers, Incorporated.
- KARGER, D., MOTWANI, R., AND SUDAN, M. 1998. Approximate graph coloring by semidefinite programming. *Journal of the ACM (JACM)* 45, 2, 246–265.
- KARP, R. M. 1972. Reducibility among combinatorial problems. *Complexity of Computer Computations*, 85–103.
- KHANNA, S., LINIAL, N., AND SAFRA, S. 2000. On the hardness of approximating the chromatic number. *Combinatorica* 20, 3, 393–415.
- KOLAITIS, P. G., PRMEL, H. J., AND ROTHSCCHILD, B. L. 1987. k_{l+1} -free graphs: Asymptotic structure and a 0 – 1 law. *Transactions of the American Mathematical Society* 303, 2, pp. 637–671.
- LEHEL, J. 1981. Generating all 4-regular planar graphs from the graph of the octahedron. *Journal of graph theory* 5, 4, 423–426.
- LEVIN, L. 1973. Universal sequential search problems. *Problemy Peredachi Informatsii* 9, 3, 115–116.
- MALAGUTI, E. AND TOTH, P. 2010. A survey on vertex coloring problems. *International Transactions in Operational Research* 17, 1, 1–34.
- MANCA, P. 1979. Generating all planer graphs regular of degree four. *Journal of graph theory* 3, 4, 357–364.
- MARTIN H., J. A. 2011. Minimal non-extensible precolorings and implicit-relations. *CoRR (Annals of Combinatorics, In review)* abs/1104.0510, 1–10.
- MCDIARMID, C. 1979. Determining the chromatic number of a graph. *SIAM J. Comput.* 8, 1, 1–14.
- MIZUNO, K. AND NISHIHARA, S. 2008. Constructive generation of very hard 3-colorability instances. *Discrete Applied Mathematics* 156, 2, 218–229.
- MULET, R., PAGNANI, A., WEIGT, M., AND ZECCHINA, R. 2002. Coloring random graphs. *Physical review letters* 89, 26, 268701.
- PARK, T. AND LEE, C. 1996. Application of the graph coloring algorithm to the frequency assignment problem. *Journal of the Operations Research Society of Japan-Keiei Kagaku* 39, 2, 258–265.
- RAMANI, A., ALOUL, F., MARKOV, I., AND SAKALLAH, K. 2004. Breaking instance-independent symmetries in exact graph coloring. In *Proceedings of the conference on Design, automation and test in Europe-Volume I*. IEEE Computer Society, 10324.
- SELMAN, B., MITCHELL, D., AND LEVESQUE, H. 1996. Generating hard satisfiability problems* 1. *Artificial intelligence* 81, 1-2, 17–29.
- STEINBERG, R. 1993. The state of the three color problem. *Annals of discrete mathematics* 55, 211–248.
- STOCKMEYER, L. 1973. Planar 3-colorability is polynomial complete. *SIGACT News* 5, 19–25.
- THOMASSEN, C. 1994. Grötzsch’s 3-color theorem and its counterparts for the torus and the projective plane. *Journal of Combinatorial Theory, Series B* 62, 2, 268 – 279.
- WANG, W.-F. AND CHEN, M. 2007. Planar graphs without 4,6,8-cycles are 3-colorable. *Science in China Series A: Mathematics* 50, 1552–1562. 10.1007/s11425-007-0106-4.
- WIGDERSON, A. 1983. Improving the performance guarantee for approximate graph coloring. *Journal of the ACM (JACM)* 30, 729–735.
- ZDEBOROVÁ, L. AND KRZAKALA, F. 2007. Phase transitions in the coloring of random graphs. *Physical Review E* 76, 3, 031131.
- ZYKOV, A. 1952. On some properties of linear complexes. *Mat. Sbornik* 24 (1949), 163,168. *American Mathematical Society Translation* 79.

Received October 2011; revised October 2011; accepted October 2011

Online Appendix to: A parametric-complexity exact 3-COLORING Algorithm

JOSE ANTONIO MARTIN H., Complutense University of Madrid

A. REPRODUCIBILITY

Working source-code of the algorithm and all the software libraries needed to properly use it and experimenting with it have been released and are available at the publishers web site.

Furthermore, there is also a web-application that implements the algorithm inside the Google App Engine cloud computing framework. The user can visit the site and test the algorithm at the following url:

— <http://graph-coloring.appspot.com>

The web coloring application just asks for a file where a graph is defined following the plain text version of the simple edge-list DIMACS standard format specification (<http://mat.gsia.cmu.edu/COLOR/general/ccformat.ps>), such as the .col files in <http://mat.gsia.cmu.edu/COLOR/instances.html>. An example of a graph file “gad-get.col” follows:

© 2011 ACM 0000-0000/2011/07-ART1 \$10.00
DOI 10.1145/0000000.0000000 <http://doi.acm.org/10.1145/0000000.0000000>

```
c generated by Jose Antonio Martin H.  
p edge 13 24  
e 1 2  
e 1 8  
e 1 9  
e 2 3  
e 2 10  
e 3 4  
e 3 10  
e 4 5  
e 4 11  
e 5 6  
e 5 11  
e 6 7  
e 6 12  
e 7 8  
e 7 12  
e 8 9  
e 9 10  
e 9 12  
e 9 13  
e 10 11  
e 10 13  
e 11 12  
e 11 13  
e 12 13
```

